Amendments to the Specification:

Please see the attached amendments as noted on the specific pages and paragraphs of each addition.

At page 1, line 1, change the paragraph to read as follows:

This application is related to a co-pending application USSN 09/658,959 entitled "Automatic Fault Management System Utilizing Electronic Service Requests", and has since issued as Patent No. 6,654,915 on 11/25/2003 which is incorporated herein by reference.

At page 1, line 6, change the paragraph to read as follows:

This application is related to a co-pending application USSN 09/731,504 entitled "Electronic Service Requests Generator For Automatic Fault Management System", and has since issued as Patent No. 6,718,439 on 04/06/2004 which is incorporated herein by reference.

At page 15, line 24, change the paragraph to read as follows:

49. Unisys-Supplied Policy File: A file of predefined XML statements describing health monitoring policies that is included with the HealthMonitor service. See "Canned XML". (Glossary #21)

At page 18, line 19, change the paragraph to read as follows:

The final piece of the solution involves two new knowledge scripts (712) that run in the Server Director (an application in the Unisys Server 702). The scripts provide notification of health events as they are detected in an environment and a form that is already familiar to existing Server Sentinel customers. These scripts retrieve the server (non-predictive) alerts and predictive alerts from the event collections maintained by the HealthEvents object and flash the associated icon in the Server Director tree view to direct the end user to the site of the problem. (Predictive alerts will flash both the icon for the affected system component and the "Predictive Alerts" icon, server alerts will flash on the icon of the affected component[[;]] [[see]] seen in Fig. 8.) Server Director also provides a rich set of additional corrective and notification actions that can be associated with either or both scripts, as the user desires.

At page 20, line 13, change the paragraph to read as follows:

FIG. 7 illustrates a generalized block diagram in which the method of the present invention could be used. A Microsoft Windows .NET operating system exists (700), which

communicates data to and from a Unisys server 702. A series of processes are included in ~~module~~ Service Module 720, which include a HealthEvents.dll file (706), which communicates with a data store 708. The data store 708 contains the PredictiveEvents and HealthEvents collections 710. The data in these collections is accessed through the HealthEvents.dll (706) hosted by the Microsoft Windows .NET operating system 700. This series of process and data in Service Module (720) receive their input from the HealthMonitor service 704. A user client application or script 712, also hosted by the Microsoft Windows .NET operating system 700, maintains communication with the set of processes in Service Module 720 as well.

At page 21, line 12, change the paragraph to read as follows:

FIG. 2 is a flowchart that illustrates the start of HealthMonitor policies. The process begins by looking at each policy "P" defined in a Unisys-supplied policy file (Block 2000). An inquiry is made at step 2006 to ~~—~~ check [[is]] P is enabled[[.]]? If P is not enabled (NO to inquiry 2006), the process returns to check the remaining policies P defined in the Unisys-Supplied Policy file back at step 2000. If the answer to inquiry 2006 is YES, and P is enabled, then for each of these policies defined, the attributes for P are read from the file to determine what to monitor, how often, and what action, if any, to take when the policy is violated (Block 2002). For example, one policy monitors the CPU utilization of a processor every 20 seconds and writes a warning message to the event log if the utilization exceeds 90%. Monitoring on P then begins on a separate processing thread T (Block 2004) initiated by the HealthMonitor service. This process can be seen in more detail in FIG. 3. The process then continues to loop back to step 2000, where the remaining policies P defined in the policy file are checked to see if they should be deployed on this system.

At page 23, line 14, change the paragraph to read as follows:

FIG. 4 is a flowchart that illustrates the process of the HealthMonitor ProviderCheck loop. First, a timer is started to run for 24 hours (Block 4000), and then for each provider X (Glossary # 36) defined in the Unisys-supplied monitoring policies (Block 4002), an inquiry is made (Diamond 4004). Inquiry 4004 checks to see if provider X is currently available. If the provider X is not available (No), Provider X is disabled (Block 4006), and the process continues at Inquiry 4010. If provider X is available (Yes), then another inquiry is made at step 4007 to

check if provider X is already enabled. If provider X is already enabled, then the process continues at Inquiry 4010. If the answer to inquiry 4007 is No, and provider X is not already enabled, provider X will be enabled at step 4008. Separate processing is started for Thread T for provider X at step 4009. Inquiry 4010 then checks if provider X is the last provider defined in the monitoring policies. If there are no more providers to check, then the timer is set to start for 24 hours again at step 4000, and the process continues; otherwise the process returns to Block 4002 to check for the next provider. The loop described by the steps in FIG. 4 allows the HealthMonitor service to detect changes in provider availability so that only those monitoring policies whose providers are enabled will be run. If a provider is added after the HealthMonitor service starts, the policies for that provider will become active the next time this loop executes; likewise, policies that have been rendered inapplicable because their provider has become unavailable will be disabled at the next loop. The loop does not terminate until the HealthMonitor service is terminated.